Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Tvorba WWW aplikace s využitím relační databáze pro zimní sporty

Michal Žalud

Bakalářská práce
2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Michal ŽALUD

Studijní program: B2646 Informační technologie

Studijní obor: Informační technologie

Název tématu: Tvorba WWW aplikace s využitím relační databáze pro půjčovnu vybavení pro zimní sporty

Zásady pro vypracování:

Úkolem bakalářské práce je vytvořit www prezentaci několika poboček půjčovny sportovního vybavení. Teoretická část se bude zabývat možnostmi zabezpečení dat uložených v databázích a využívaných WWW servery před neoprávněnými přístupy prostřednictvím Internetu. Aplikace musí minimálně umožnit: * Registraci uživatelů * On-line rezervace výpůjček * Realizace rezervovaných výpůjček na půjčovnách * Půjčování vybavení přímo na půjčovnách bez předchozí rezervace * Záznamy o půjčeném sportovním vybavení * Fakturace výpůjček * Přístup dle práv

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: 2008/2009

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Michal ŽALUD

Studijní program: B2646 Informační technologie

Studijní obor: Informační technologie

Název tématu: Tvorba WWW aplikace s využitím relační databáze pro půjčovnu vybavení pro zimní sporty

Zásady pro vypracování:

Úkolem bakalářské práce je vytvořit www prezentaci několika poboček půjčovny sportovního vybavení. Teoretická část se bude zabývat možnostmi zabezpečení dat uložených v databázích a využívaných WWW servery před neoprávněnými přístupy prostřednictvím Internetu. Aplikace musí minimálně umožnit: * Registraci uživatelů * On-line rezervace výpůjček * Realizace rezervovaných výpůjček na půjčovnách * Půjčování vybavení přímo na půjčovnách bez předchozí rezervace * Záznamy o půjčeném sportovním vybavení * Fakturace výpůjček * Přístup dle práv

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č.121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 13. 8. 2009

Michal Žalud

Souhrn

Cílem projektu bylo vytvořit webovou aplikaci pro rezervování zimního vybavení v malém horském středisku. Aplikace pracuje na databázovém systému Oracle a využívá jazyků PHP,HTML a SQL. Grafické prostředí je formátováno jazykem CSS.

Práce se dále zabývá bezpečností webové aplikace, útoky na ní a způsoby jak se jim bránit.

Klíčová slova

rezervace, PHP, HTML, CSS, SQL

Title

Creating web applications using a relational database for the rental of winter sports

Abstract

The aim of the project was to create a web application for reservation of winter equipment in a small mountain resort. The application works on the Oracle database and uses the languages PHP, HTML and SQL. Graphical environment is CSS formatting language.

This work also addresses the safety of web applications, attacks on it and ways to prevent them.

Keywords

reservation, PHP, HTML, CSS, SQL

Obsah

Seznam obrázku

Seznam použitých odborných výrazů

Výraz	**Popis**
HTML	Hypertext Transfer Protocol, internetový protokol
CSS	Cascading Style Sheets, jazyk pro formátování stránek
PHP	Hypertextový preprocesor, skriptovací programovací jazyk
Apache	nejrozšířenější webový server
SJSWS	Sun Java System Web Serveru
XHTML	Extensible Hypertext Markup Language
SQL	Structured Query Language, dotazovací jazyk
DDL skript	skript pro definici dat
primární klíč	pole nebo kombinace polí, jednoznačně identifikující každý záznam v databázové tabulce.
cookies	data zaslaná serverem uložená na straně klienta
WYSIWYG	What you see is what you get, česky „co vidíš, to dostaneš“, způsob editace dokumentu
Oracle XE	Oracle Database Express Edition, verze databázového serveru Oraclu, jenž je dostupná zdarma

1. Úvod

Cílem bakalářské práce je vytvořit webovou aplikaci sloužící několika pobočkám zimního vybavení. Aplikace je určená v prvé řadě pro zákazníky, kterým slouží k vytvoření rezervace zimního vybavení. Dále aplikace poskytuje funkce nezbytné pro chod poboček. Mezi nezbytné funkce patří spravování poboček, vybavení a uživatelů. Hlavní využití aplikace je zdokonalení služeb půjčovny, především se jedná o rezervaci vybavení, jenž značně ušetří čas jak zákazníkům tak zaměstnancům.

V první části se práce věnuje základním útokům vedených proti webovým aplikacím. Bude ukázáno jak útoky mohou nastat a kde se vyskytují.

V další části práce bude provedena analýza aplikace se stanovením cílů. Dále se věnuje návrhu databáze a použitým databázovým objektům. V této části je také uveden stručný popis tabulek databáze.

Poslední část je věnována použitým technologiím při vytváření aplikace. Také jsou zde uvedeny konstrukce využité v aplikaci.

2. Útoky na webové aplikace

„Bezpečnost nadevše“, toto tvrzení platí nejen v životě, ale dá se použít i pro webové aplikace. Bezpečnost se dá považovat za nejdůležitější část při vytváření aplikace. Asi nikomu by se nelíbilo, kdyby někdo manipuloval s jeho daty případně disponoval osobními údaji zadanými například při registraci. Webová aplikace většinou využívá webového a databázového serveru. Aplikační logika je napsaná v jednom či více programovacích jazycích. Pro vytvoření bezpečné aplikaci je nutné dodržovat velké množství pravidel a doporučení, jedná se o kombinaci dobře napsaného kódu aplikace, správného nastavení webového a databázového serveru. Existuje celá řada útoků na webové aplikace. Jejich typy a možnosti jsou svázány se samotným vývojem webových aplikací a technologiemi, které aplikace používají.

2.1. Cross Site Scripting

Cross Site Scripting je výraz používající se pro útoky jenž spočívají ve vložení nebezpečného kódu do webové stránky. Kód je napsán ve skriptovacím jazyku, jedná se především o Javascript, lze ale využít i dalších jazyků. K vložení kódu se převážně využívá neošetřených vstupů od uživatele, jedná se o formuláře či příspěvky v diskusích fórech. Útok Cross Site Scripting též označovaný jako XSS lze rozdělit do tří typů: okamžitý, perzistentní a lokální.

Okamžitý

Jak už název napovídá jedná se o útok, který spočívá v okamžitém vykonání vloženého skriptu. Jedná se především o webové aplikace, které zobrazují výsledek okamžitě na základě požadavků. Může se zdát, že se jedná o triviální problém, ale není tomu tak. Většina aplikací využívá parametrů z URL adresy, toho může útočník využít tím, že do adresy přidá vlastní nebezpečný kód. Vložený kód není upravován, takže se prohlížeč zachová,jakoby vložený kód patřil do naší stránky. Tohoto se využívá hlavně u generovaných stránek například stránek využívajících PHP.

Následující kód názorně ukazuje okamžitý XSS útok. Pokud by se kód provedl zobrazilo by se nám výstražné okno s nápisem „Toto je úspěšný XSS útok.“.

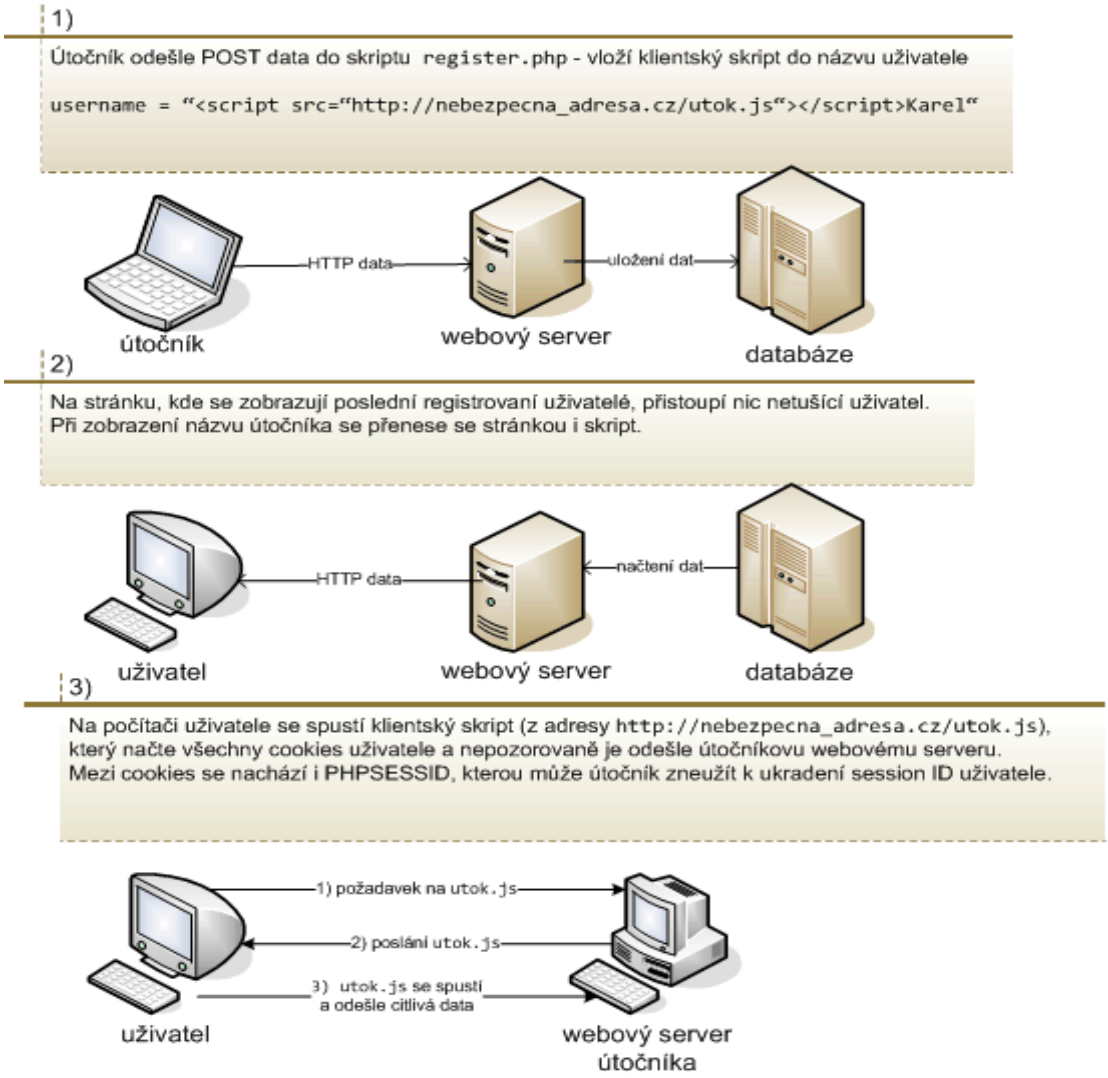
```
<?php echo $_GET['nadpis']; ?>
http://URL/stranka.php?nadpis=nebezpecny_kod<script>alert
('Toto je úspěšný XSS útok.');</script>
```

Lokální

Tento typ útoku bývá označován jako DOM-based či local. Jedná se do určité míry o okamžitý útok. Rozdíl spočívá v tom, že útok je veden na statické stránky na kterých se vyskytuje nějaký klientský skript. Skriptu se podstrčí proměnná, která může obsahovat nebezpečný kód. Hlavní nebezpeční tkví v neomezeném přístupu klientských skriptů na lokální disk.

Perzistentní

Zdaleka nejnebezpečnější XSS útok je označován jako perzistentní. Využívá místa aplikace, kde uživatel předává aplikaci vlastní obsah. Může se jednat například o příspěvky do diskusního fóra, kde se namísto příspěvku vloží nebezpečný skript. Při zobrazování příspěvků se kromě standardních příspěvků načte i skript, který může způsobit nedozírné škody. Útok se nemusí týkat jen diskusních fór, ale útočící skript se může vložit i do vstupního pole formuláře, například uživatelského jména. Pokud si někdo bude prohlížet registrované uživatele, načte se mu spolu se jmény i skript. Dá se říci, že perzistentní XSS útok se může vyskytovat u aplikací, které generují obsah stránky z databáze. Typický příklad perzistentního XSS útoku je demonstrován na obr. 1.



obr. 1 - perzistentní XSS útok (7)

2.2. Cross-Site Request Forgery

CSRF útok by se dal česky nazvat „ podvržení požadavku mezi různými stránkami“. Někdy se můžeme setkat s termíny Cross-Site Reference Forgery, Session Riding. S tímto útokem se můžeme setkat především u aplikací, které využívají cookie, typicky se jedná o session proměnné v jazyce PHP. Útok bývá veden především proti aplikacím do, kterých se útočník může sám přihlásit a tím získat představu o jejich struktuře.

Jelikož je http protokol bezstavový a tudíž není schopen si nic zapamatovat, je nutné využívat jiné metody pro uchovávání nezbytných informací jako například informace o identitě návštěvníka. Využívají se tedy cookie, typicky se jedná o session proměnné v jazyce PHP. Samotný útok je založen na tom, že uživatele jenž je přihlášen do nějaké aplikace,přimějeme k návštěvě napadené stránky. Stačí kliknutí na normálně vyhlížející odkaz, jenž ale vede na napadené stránky. Některé útoky ani nepotřebují, aby uživatel přešel na jiné stránky, stačí předem připravený skript, který se vykoná po určité události. Vše má za cíl odcizení těchto cookies a tím získání identity nic netušícího uživatele.

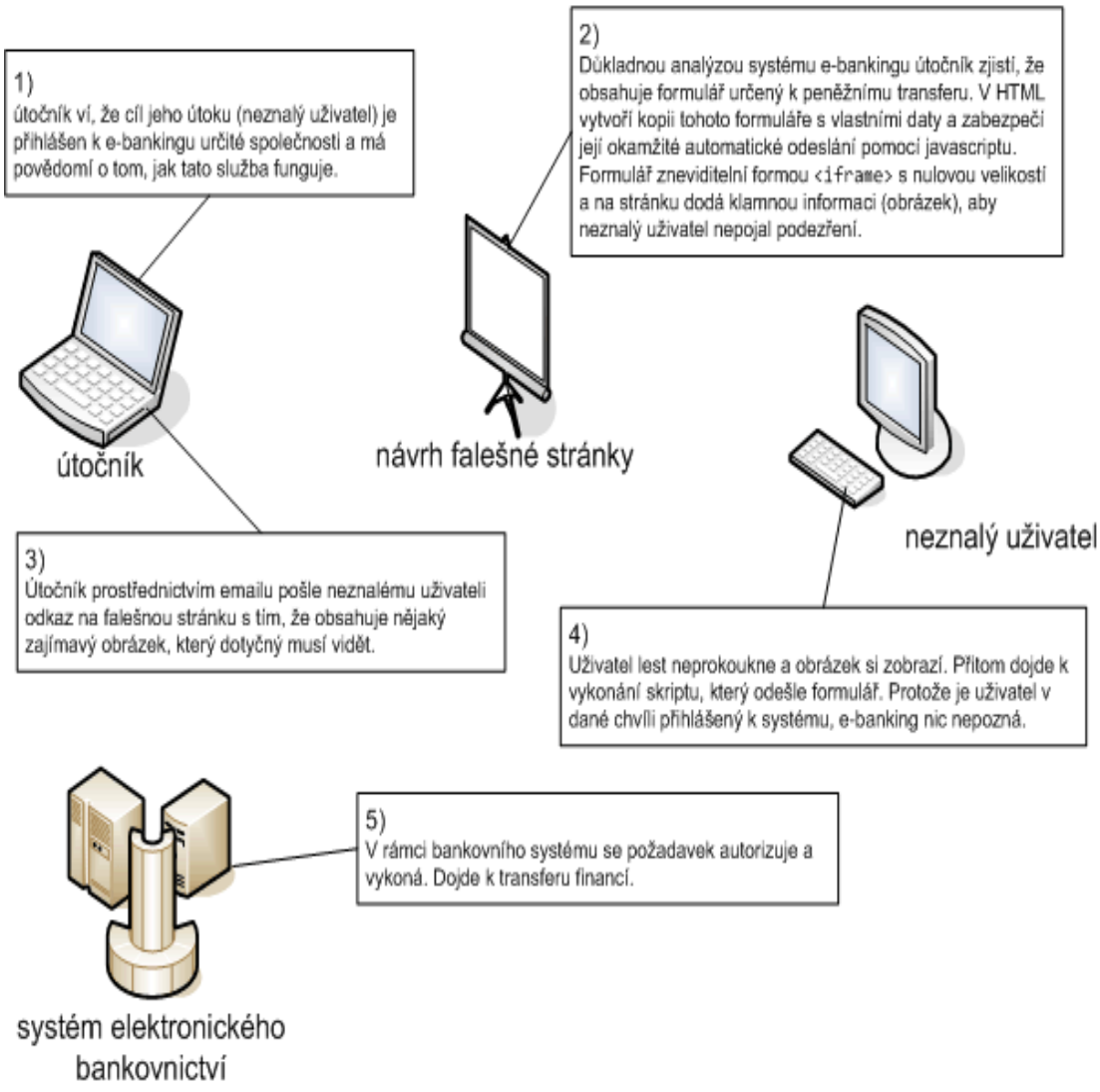
Samotné navštívení nebezpečné stránky a s tím spojené zjištění indetity je určitě nebezpečné, ale nedá se považovat za útok. Útok nastává v okamžiku, kdy útočník pod získanou identitou začne provádět určité akce - útoky. Obvykle se k vykonání útoku využívá metod GET a POST, jedná se o metody, jenž předávají serveru data.

CSRF útok pomocí metody GET je velice prostý a o to více nebezpečnější. Základem útoku je opět donutit útočníka navštívit útočníkovu stránku, na které je umístěn tag img, jenž slouží pro zobrazování obrázku, a jeho parametru src, do kterého se zadá formulář, jenž simuluje formulář na napadené stránce. Při načtení útočníkovi stránky se vykoná úkon uvedený v parametru src u “obrázku“. Důležité je si uvědomit, že útok může být proveden pouze v případě, že jsme přihlášeni do napadené aplikace.

Následující kód ukazuje příklad využití obrázku pro CSRF útok na bankovní účet.

```
<img
src="http://www.vasebanka.cz/submitTranser/?account=1/123
4&amount=1000000&confirm=1">
```

Metoda POST je bezpečnější, protože nepředává své parametry pomocí URL adresy. Ovšem i aplikace využívající výhradně metody POST jsou napadnutelné útokem CSRF. Především se využívá zapnutého JavaScriptu. Může se, ale využít i předvyplněného formuláře, kdy přesvědčíme uživatele, že je důležité formulář odeslal na server. Ve formuláři ovšem uvedeme námi zvolené hodnoty. Na obr. 2 je uveden příklad CSRF útoku na bankovní služby.



obr. 2 - CSRF útok na e-banking (7)

2.3. PHP injection

Tento problém se týká opravdu začátečníků. Útok pomocí PHP injection je založen na vložení a vykonání útočníkova skriptu. K vložení se využívá nesprávné použití funkcí *include* a *require*. Tyto funkce pracují tak, že na místo svého výskytu vloží soubor uvedený jako parametr této funkce. Funkce *include* a *require* nacházejí uplatnění především na úvodních stránkách jako je index.php. Využívají se především pro zjednodušení práce, protože nemusíme na každou stránku psát záhlaví, patičku atd.

Případný útok může vypadat následovně, kdy se nám vykoná zakernyskript.php.

```
index.php?stranka=http://domena.cz/zakernyskript.php
```

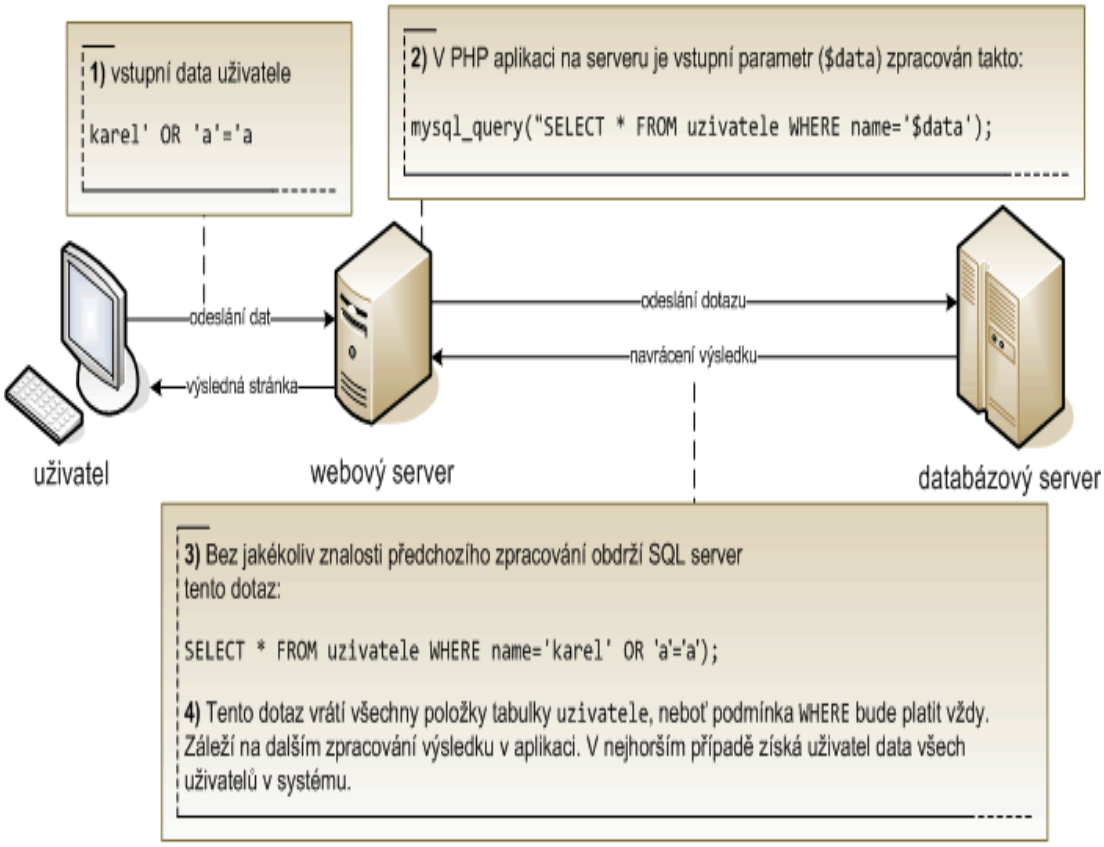
Obrana proti tomuto útoku je velice jednoduchá, jedná se o nadefinování podmínek pro vkládání všech stránek aplikace.

2.4. SQL injection

Základem většiny webových aplikací je databáze, ve které jsou uložena potřebná data. K získávání dat se využívá SQL dotazu. Právě pozměnění původního dotazu se nazývá SQL injection.

Tímto útokem mohou být napadeny všechny webové aplikace, ve kterých není integrována dostatečná kontrola informací předávaných od uživatele. K útoku se může použít například parametrů v URL či data předávaná formulářem.

Mezi možné dopady úspěšné modifikace SQL dotazu může patřit získání dat, ke kterým bychom neměli mít přístup. Podstatně nebezpečnější je, pokud útočník může upravovat data v databázi. V takovém případě mu nic nebrání ve smazání tabulek či dokonce celé databáze. Pro ukázku celého problému poslouží obr 3.



obr. 3 - příklad SQL injection (7)

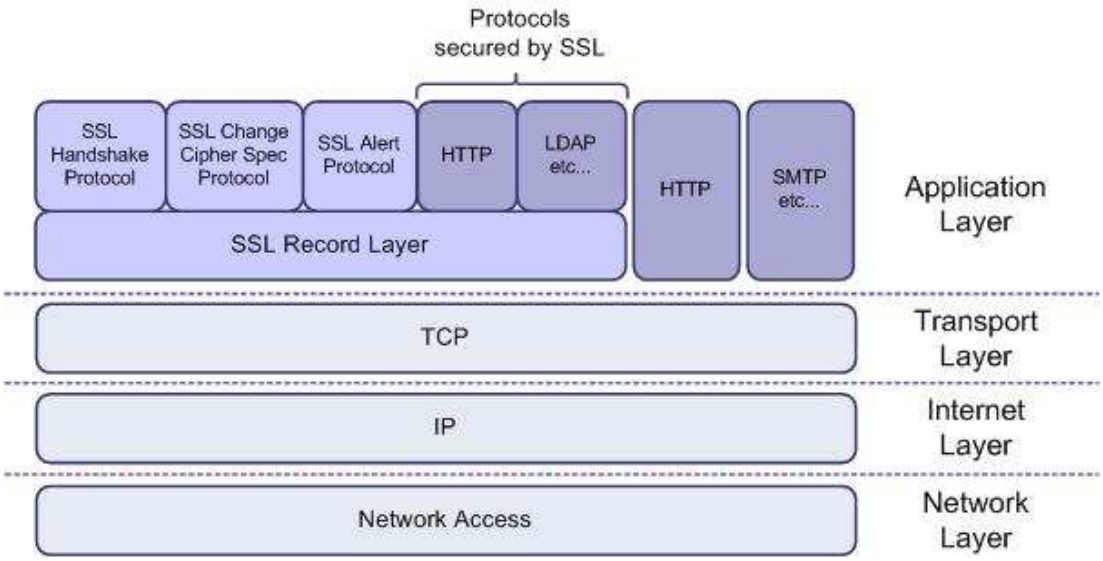
3. Bezpečnější Apache

Každá úprava vedoucí k zvýšení bezpečnosti webové aplikace je užitečná. Útočníkovi určitě pomůže, když ví, na koho útočí, proto je dobré nedávat každému přístup k informacím o operačním systému či konfiguraci webového serveru. K získání informací slouží direktivy `ServerSignature` a `ServerTokens`. Druhá direktiva určuje stupeň podrobnosti výpisu první, je vhodné ji nastavit na `Prod`, čímž se nám do HTTP hlavičky vloží pouze informace, o jaký webový server se jedná. Další informace jako verze serveru, operační systém na kterém server běží, se do výpisu neuvádějí.

Určitě není dobré, když se uživateli zobrazí kompletní souborová struktura v případě, že požadovaná stránka nebyla na serveru nalezena. Toho se dosáhne nastavením `Options` u direktivy `DocumentRoot` a sice nesmí být v nastavení uvedeno `Indexes`, které nám způsobuje zmiňované indexování. Apache můžeme obohacovat o celou řadu modulů, proto se doporučuje zapnout pouze ty námi požadované.

3.1. Protokol SSL

Protokol SSL zajišťuje šifrování dat, autentizaci serveru a vytváří šifrované spojení. Z pohledu protokolu TCP/IP se jeví jako vrstva, jenž je vložena mezi transportní a aplikační vrstvu. Tato vrstva umožňuje vytvořit šifrované spojení mezi dvěma komunikujícími stranami. U webových aplikací se obvykle svou totožností prokazuje pouze server. K autentizaci se využívá asymetrického šifrování a certifikátů, které vydává certifikační autorita. Pravost certifikátu lze ověřit pomocí veřejného klíče certifikační autority.



obr. 4 - začlenění SSL protokolu (5)

4. Bezpečnější PHP

Správné nastavení PHP je prvním a velice zásadním krokem k vybudování bezpečné aplikace. V dnešní době běží většina aplikací na serveru, kde se o místo dělí více uživatelů, v tomto případě je dobré mít zapnutý bezpečný mód (safe mode), který má za úkol zamezit spouštění skriptů, neoprávněnému prohlížení a modifikaci souborů. Tento mód není všemocný, je také nutné mít správně nastavená práva u souborů.

4.1. Safe mode

Bezpečný mód zapneme tak, že v souboru php.ini nastavíme safe_mode na ON. Způsob kontroly je založen na ověřování identity uživatele spuštěného skriptu a identity uživatele vlastnícího soubor, jenž chce skript použít. V případě shody identity je vše v pořádku, v opačném případě mu přístup nebude umožněn. V některých situacích mohou s použitím výchozího nastavení nastat problémy, proto je možné pomocí parametru chování safe_mode upravit.

Typickým příkladem je situace, kdy si chce uživatel Karel přečíst soubor, který se jmenuje pepuv_soubor a vlastní ho uživatel Pepa, skript přistupující k souboru pepuv_soubor vlastní Karel, ale vlastníkem souboru je Pepa. Zde zafunguje safe_mode a neumožní Karlovi, jakožto vlastníku skriptu přistupujícího k souboru, který vlastní Pepa, aby přistoupil k souboru pepuv_soubor. Aby Karel mohl přistupovat k souborům, které vlastní Pepa, je možno využít direktivu *safe_mode_grid*, která místo vlastníka skriptu a souboru ověřuje, zda jsou ve stejné skupině. Další zajímavou direktivou je *safe_mode_exec_dir*, ta slouží k určení umístění systémových programů, které mohou být spuštěny funkcemi system() či exec().

4.2. Direktivy nastavení PHP

Bezpečný mód není zdaleka jediné nastavení, které je dobré uplatnit, aby naše PHP bylo bezpečnější. Existuje celá řada direktiv, o jejichž nastavení je nanejvýš vhodné popřemýšlet vzhledem k bezpečnosti.

Disabled_function je direktiva, pomocí které můžeme zakázat vybrané funkce. Výše zmíněný problém s otevíráním souboru lze také vyřešit nastavením *disable_function = fopen*.

V blízké budoucnosti se pravděpodobně ve velké míře začne využívat direktiva *disable_classes*, jenž je spojená s objektově orientovaným přístupem a zakázáním určitých tříd.

Direktiva *max_execution_time* specifikuje maximální čas běhu skriptu, většinou je nastaven na 30 sekund. Direktiva tak ochraňuje server před vyčerpáním prostředků. Nebezpečné je její nastavení na nulu, tím není čas nikterak omezen. Direktiva *memory_limit* udává maximální velikost paměti, kterou skript může využít.

K zamezení neoprávněnému prohlížení souborů lze využít direktivu *open_basedir*, ta určí místa na serveru, kam má uživatel přístup. Její funkce je podobná direktivě DocumentRoot u webového serveru Apache.

Aby se skripty na serveru “povalovaly“, kde se uživateli zlíbí, není určitě správné. Určení místa pro ukládaní skriptů udává direktiva *user_dir*, ve většině případů se používá ve spolupráci s direktivou UserDir u webového serveru Apache.

Určitě není dobré dávat informace o konfiguraci PHP k nahlédnutí každému, kdo má o to zájem. Všichni zájemci o informace, například o operačním systému, o verzi PHP či o verzi databázového serveru, jsou potencionální útočníci na náš server. Všechny tyto a mnoho dalších informací nám poskytuje funkce *phpinfo*, je proto nanejvýš vhodné tuto funkci uvést do výčtu zakázaných funkcí u direktivy *disable_function*.

5. Bezpečnost databázového serveru

Pojem bezpečnost se u databázového serveru dá chápat z několika pohledů. Může jít o poškození či ztrátu dat, riziko se snižuje vhodným zálohováním. Druhý pohled rizika tkví v neoprávněném přístupu k datům, k minimalizaci tohoto problému se využívá správně navrhnutých přístupových práv k databázovému serveru.

5.1. Zálohování dat

O významech zálohování dat dnes nepochybuje snad nikdo. Stačí si připomenout tragický teroristický útok na mrakodrapy Světového obchodního centra v New Yorku, kdy pravděpodobně druhou největší ztrátou (hned po lidských životech) byla ztráta dat.[1]

Důležité je určení způsobu, jak zálohovat data určitého typu. Data uchovávající diagnostické záznamy či informaci o provozu určité technologie stačí ukládat na sekundární disk počítače či na jiný počítač, nacházející se v blízkosti zálohovaného. Ztráta těchto dat není příjemná, ale katastrofální dopady to obvykle mít nebude. Záznamy citlivé na případnou ztrátu je vhodné zálohovat na jiné místo než původní záznamy. Může to být například i na druhé straně světa.

Způsob zálohování lze rozdělit do tří typů

- kompletní zálohování je způsob, při němž se zálohují všechna data i databázové struktury. Je patrné, že nevýhodou je nutnost zálohovat velké množství dat, výhodu lze spatřit v jednoduché obnově ze záložní databáze.

- způsob, kde se neuchovávají všechna data, ale pouze rozdíly v záložní a zálohované databázi se nazývá rozdílové zálohování. Výhodou je mnohem menší objem dat při záloze, kterou je ale potřeba provádět častěji než kompletní zálohu

- zálohování transakčního žurnálu spočívá v zálohování transakcí prováděných od poslední kompletní zálohy. Transakčních žurnálů se obvykle využívá více, jejich uchovávání může vyvolat uživatel nebo se děje automaticky.

Zálohování je možné rozdělit také podle času aktualizace zálohy

- Synchronní aktualizace dat spočívá v souběžném zapisování dat do zálohované a záložní databáze.

- Aktualizace založená na zpoždění zápisu dat se nazývá asynchronní, její výhody se mohou projevit při selhání některé operace. Minimalizuje dopad nejčastějších důvodů k zálohování dat a sice chyby způsobené uživatelem.

5.2. Řízení přístupu

Proč řídit přístup a využívat další bezpečnostní mechanismy na úrovni databázového serveru, když to lze provádět i na úrovni vlastní aplikace?

- *Pokud k datům přistupuje více aplikací, jsou práva definována centrálně a není třeba je implementovat opakovaně v každé aplikaci. To snižuje riziko, že některá z implementací nebude dostatečně kvalitní.*

- *Řízení přístupu i řada dalších bezpečnostních mechanismů je převážně deklarativní, nevyžadujících programování. Lze tedy relativně jednoduše pomocí systémových pohledů kontrolovat správné nastavení. Oproti tomu kontrolovat správnost řízení přístupu v kódu programu je značně problematické.*

- *Dochází k minimalizaci rozsahu možného ohrožení dat v případě, že uživatel získá kontrolu nad aplikací samotnou, například tolik populární napadení pomocí SQL Injection může mít nulové, nebo jen minimální následky, pokud aplikace k databázi přistupuje přes uživatele s minimálními právy, která dovolují provést na úrovni databáze jen ty operace, které lze provést i přes aplikaci. Pokud ale aplikace přistupuje k databázi s právy vlastníka databázových tabulek nebo dokonce databázového administrátora, jsou vaše data nechána na pospas útočníkovi.*

- *Na rozdíl od aplikace vyvíjené na míru pro jednoho zákazníka, je databázový server software, provozován stovkami tisíc zákazníků po celém světě. Bezpečnostní mechanismy databázového serveru jsou tak prověřené mnohaletým provozem v řadě značně rozdílných prostředí.* [3]

5.2.1. Uživatelská oprávnění

Je velmi důležité správně definovat oprávnění jednotlivých uživatelů. Může jít o práva nad systémem nebo práva vztahující se k objektům. Určitě není vhodné, aby každý uživatel, jenž přistupuje k databázi, vlastnil práva na mazání, vytváření tabulek či dokonce administrační oprávnění. Proto platí pravidlo přidělování minimálních nutných práv.

Systémová oprávnění slouží k zabezpečení přístupu k databázovému serveru a jeho správu. Oprávnění k provádění systémových úkonů jednotlivým uživatelům přiděluje administrátor. Ten má obvykle oprávnění k vytváření, mazání uživatelů, odstraňování tabulek a samozřejmě práce s právy uživatelů.

```
GRANT oprávnění, oprávnění 2, … TO uživatel;
// příkaz pro přidělení práv
REVOKE oprávnění, oprávnění 2, … FROM uživatel;
// příkaz pro odebrání práv
```

Databáze je tvořena tabulkami, pohledy, sekvencemi a mnoha dalšími objekty, každý objekt má svého vlastníka, který má nad daným objektem právo přidělovat oprávnění ostatním uživatelům.

```
GRANT oprávnění, oprávnění 2, … ON objekt TO uživatel;
// příkaz pro přidělení práv nad objektem
REVOKE oprávnění, oprávnění 2, … ON objekt FROM uživatel;
// příkaz pro odebrání práv u objektu
```

U složitějších projektů, kde se vyskytuje více uživatelů se stejnými právy, se vyplatí vytvořit role a podle typu uživatele mu určitou roli přiřadit. Uživatele tím vlastně rozdělíme do skupin, stejně jako tomu bývá v reálném světě. Pro vytvoření rolí je nutné vlastnit dostatečná práva (administrátor databáze).

```
CREATE ROLE název_role;
// příkaz pro vytvoření role
GRANT oprávnění, opravnění 2, … TO název_role;
// příkaz na přidělení práv roli
GRANT název_role to uživatel;
// příkaz pro přidělení role uživateli
```

K jedné roli může patřit více uživatelů,to je z jejího určení zřejmé, ale i k jednomu uživateli může patřit více rolí. Administrátorovi ulehčí práci profily, což se dá chápat jako výčet omezení. Umožňují například nastavit platnosti hesel či počet pokusů k přihlášení do databáze. K vytvoření profilu je samozřejmě potřeba mít dostatečná oprávnění.

```
CREATE PROFILE název_profilu LIMIT
FAILED_LOGIN_ATTEMPTS
PASSWORD_LIFE_TIME
CONNECT_TIME 480;
```

5.2.2. Nástroje poskytující Oracle

Oracle nabízí užitečný nástroj pro správu přístupů, jenž umožňuje přistupovat uživateli pouze z určité aplikace, například není povolen přístup z databázové konzole. Nástroj se jmenuje Secure Application Role, využívá rolí, které se vážou na konkrétní databázový balík, který lze chápat jako zapouzdření databázových objektů jako jsou procedury, funkce či triggery.

Ve větších firmách se určitě vyplatí zvážit použití nástroje Virtual Private Database (VPD), pracuje tak, že dotaz doplní o podmínku, ta omezuje data, s kterými může uživatel pracovat. Typickým příkladem jeho využití je poskytování informací jednotlivým oddělením. Uživatelé ze dvou oddělení vznesou stejný dotaz, ale pro každého se vyhodnotí jinak, právě díky doplnění podmínky.

Zaznamenávání informací o databázových operacích lze označit jako audit databáze. Audit slouží k dodatečnému zjištění, jaké operace jednotliví uživatelé prováděli, lze tedy zjistit případné neautorizované aktivity. Mechanismy pro vytváření auditů obvykle uchovávají velké množství záznamů, ve kterých najít nebezpečné operace není jednoduché. Za účelem zmenšení počtu záznamů Oracle obsahuje tzv. Fine Grained Auditing, jeho funkce spočívá v konkretizaci dat, u kterých se má audit provádět. Pokud operace začne pracovat s definovanými daty bude zaznamenána. Při využití mechanismu Flashback je dokonce možné získat výsledek z doby, kdy byla operace provedena. Mechanismus Flashback si totiž může vyžádat data z určitého bodu v minulosti.

6. Analýza aplikace

V posledních letech stále více lidí tráví v zimním období volné chvíle na sjezdových svazích či běžeckých tratích. Ne všichni ovšem mají vlastní lyžařskou výbavu. Pro někoho je to příliš finančně náročné, jiný chce mít každý rok to nejmodernější vybavení. Jsou i tací, kteří si prostě jízdu na lyžích či snowboardu chtějí jen vyzkoušet. Proto se i v našich horských centrech rozrostly sítě půjčoven zimního vybavení, které poskytují své služby velkému množství lidí, aby si i oni mohli vychutnat zimní radovánky podle svých přestav. Při velkém počtu, lyží, snowboardů a dalšího vybavení, je důležité, ale přitom velmi složité, mít o něm přehled. Nejde jen o zjištění aktuálně půjčeného zboží, ale i plánování servisní údržby po jeho navrácení. Provozovatel potřebuje mít pro zefektivnění své činnosti statistický přehled o důležitých informacích, např. znát nejvytíženější pobočku, nejpůjčovanější druh zboží atd. A právě kvůli nutnosti uchovávání poměrně velkého množství informací je ideální ukládat je v databázi.

V dnešní hektické době, kdy se snažíme, co nejefektivněji využívat čas, je velice vítanou možností rezervovat si požadované vybavení už před příjezdem do lyžařského střediska. Zde si na půjčovně, již bez zbytečné časové ztráty, může zájemce vybavení vyzvednout a provést samozřejmě finanční úhradu za poskytnuté služby. Tato služba je výhodná pro zákazníka v mnoha směrech, minimalizuje se mu případná čekací doba v půjčovně a má jistotu, že jím požadované vybavené bude k dispozici v očekávané kvalitě a rozsahu. Hlavní výhoda pro majitele půjčovny spočívá v tom, že může vylepšit a urychlit poskytované služby a tím získat spokojeného zákazníka.

6.1. Cíle aplikace

Základním cílem aplikace je on-line rezervační systém pro firmu vlastnící několik poboček půjčoven zimního vybavení. Po důkladné analýze problému půjčování vybavení byly požadavky na aplikaci rozšířeny o funkce určené k spravování vybavení, poboček a samozřejmě i uživatelů.

6.1.1. Rezervace vybavení

Rezervace musí mít svého konkrétního vlastníka, aby bylo zřejmé, kdo si vybrané vybavení vyzvedne, z tohoto důvodu je třeba se do rezervačního systému přihlašovat. Zákazník, který není ještě registrován, musí vyplnit požadované údaje a zaregistrovat se.

Po úspěšné registraci a následném přihlášení už nic nebrání zákazníkovi ve vybírání vybavení dle svého přání, důležité je také, jestli vybrané vybavení bude ve zvoleném termínu dostupné. On-line rezervace je určená pro registrovaného zákazníka, který se tak vyhne frontám v půjčovně.

6.1.2. Ostatní služby a funkce půjčovny

Mezi ostatní nezbytné funkce, které by měla aplikace pro půjčovnu vybavení nabízet, lze zařadit.

- rezervace a půjčení vybavení na pobočce
- správu poboček
- správu uživatelů
- správu rezervací

Při praktickém provozu by se určitě dala najít celá řada dalších užitečných a zajímavých funkcí či služeb jako například informace o počasí. Já jsem do své aplikace zařadil možnost komunikace firmy se zákazníkem či jen návštěvníkem webových stránek, prostřednictvím aktualit. Aktuality zadává správce kteréko-li pobočky a jedná se o stručné informace, například o probíhajících akcích ve středisku, cenových akcích a podobně.

6.2. Uživatelé aplikace

V aplikaci se vyskytují čtyři role.

- zákazník
- opravář
- zaměstnanec na přepážce
- správce pobočky

Zákazník má možnost rezervovat si vybavení, možnost získat přehled o svých rezervacích a také úpravu svých registračních údajů. K rezervaci je potřeba vybrat počáteční a koncový den rezervace. Po vybrání dnů rezervace musí určit pobočku, na které si vybavení vyzvedne. Nejdůležitější je určit, o jaké vybavení a v jaké velikosti má zájem. Může se stát, že si zákazník bude chtít půjčit od jednoho druhu a jedné velikosti více kusů, proto je třeba vybrat počet vybavení. Po zvolení počtu je zákazník informován o detailu rezervace a o vybraném vybavení, které je ještě potřeba vložit do rezervačního košíku, s kterým lze dále libovolně pracovat. Zákazník má samozřejmě možnost prohlížet si své rezervace a také je zrušit, ovšem pouze do předchozího dne počátku rezervace.

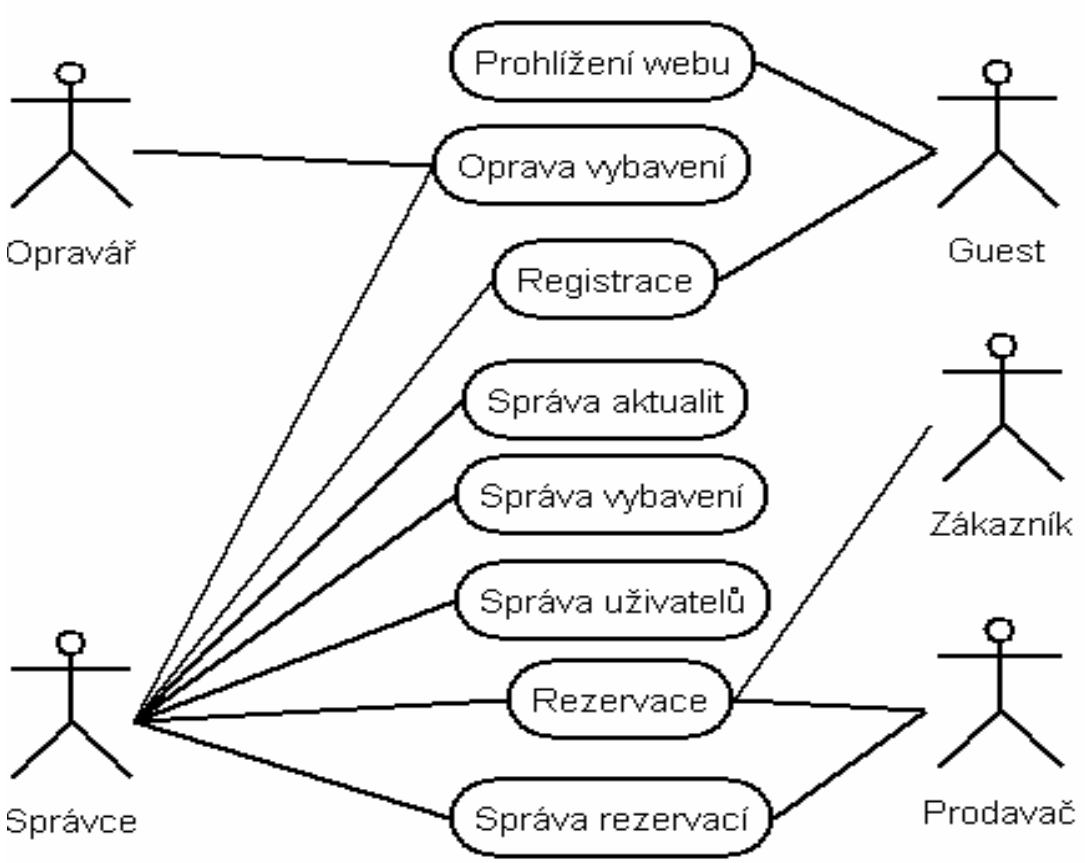
Serviceman je zaměstnanec, jehož jedinou, ovšem neméně důležitou, prací je opravování vybavení. Prodavač musí při navrácení označit vybavení, které potřebuje opravit. Označení je důležité, hlavně z důvodů ušetření času “ servisáka “ , který si může bez předchozího zkoumání, vzít rovnou poškozené vybavení a opravit jej a zavést do systému, pokud se oprava zdařila a vybavení je možno dále půjčovat.

Prodavač má za úkol rezervace nejen vytvářet, ale půjčovat vybavení přímo na pobočce. Po příchodu zákazníka s rezervací vybere určené vybavení a předá je zákazníkovi a také zadá do systému, že daná rezervace byla vyzvednuta. Platba obvykle probíhá před samotným předáním vybavení. Při navrácení rezervovaného vybavení musí prodavač zkontrolovat stav jednotlivého vybavení. V případě nutnosti opravy to zanese do systému, aby “servisák“ věděl, které vybavení je potřeba opravit.

Správce pobočky je uživatel, který má na starosti chod pobočky. Je zodpovědný za přijímání zaměstnanců, může tedy do systému zadávat nové prodavače,“ servisáky “, ale i další správce. Jeho další pravomoci jsou důležité případě, že stávající uživatelé zapomenou své heslo pro přihlášení, jedná se o možnost změny všech osobních údajů, a především pak hesla. Uživatele, kteří učinili kroky vedoucí k rozvázání spolupráce, může ze systému dočasně vyřadit tím, že jim přiřadí status neaktivní. Pokud by se prohřešky nadále opakovaly, může uživatele úplně odstranit. Správce také spravuje vybavení, může tedy do systému přidávat nové druhy a nové vybavení. Taktéž může přidávat nové pobočky. Může se stát, že technické problémy donutí uzavřít některou z poboček. Proto musí být správce schopen pobočku vyřadit z nabídky. Pokud by správce potřeboval sdělit informace zákazníkům či zaměstnancům, může k tomu využít krátkých zmíněných aktualit. Mimo všech zmíněných funkcí má správce funkcionality jako opravář a zaměstnanec na pobočce. Je to především z důvodu, že správce musí vědět o každé činnosti na pobočce vše a tudíž to i umět vykonávat.

6.2.1. Use case diagram

Use case diagram je funkční zobrazení aplikace z pohledu uživatele. Slouží k získán přehledu o možnostech aplikace, který uživatel a jaké funkce může provádět.



obr. 5 - use case diagram

6.3. Grafické prostředí aplikace

Vzhled aplikace byl volen tak, aby bylo na první pohled jasné, že se jedná o půjčovnu zimního vybavení. K tomuto účelu posloužila hlavička – místo na kterém obvykle spočine první pohled návštěvníka stránky.

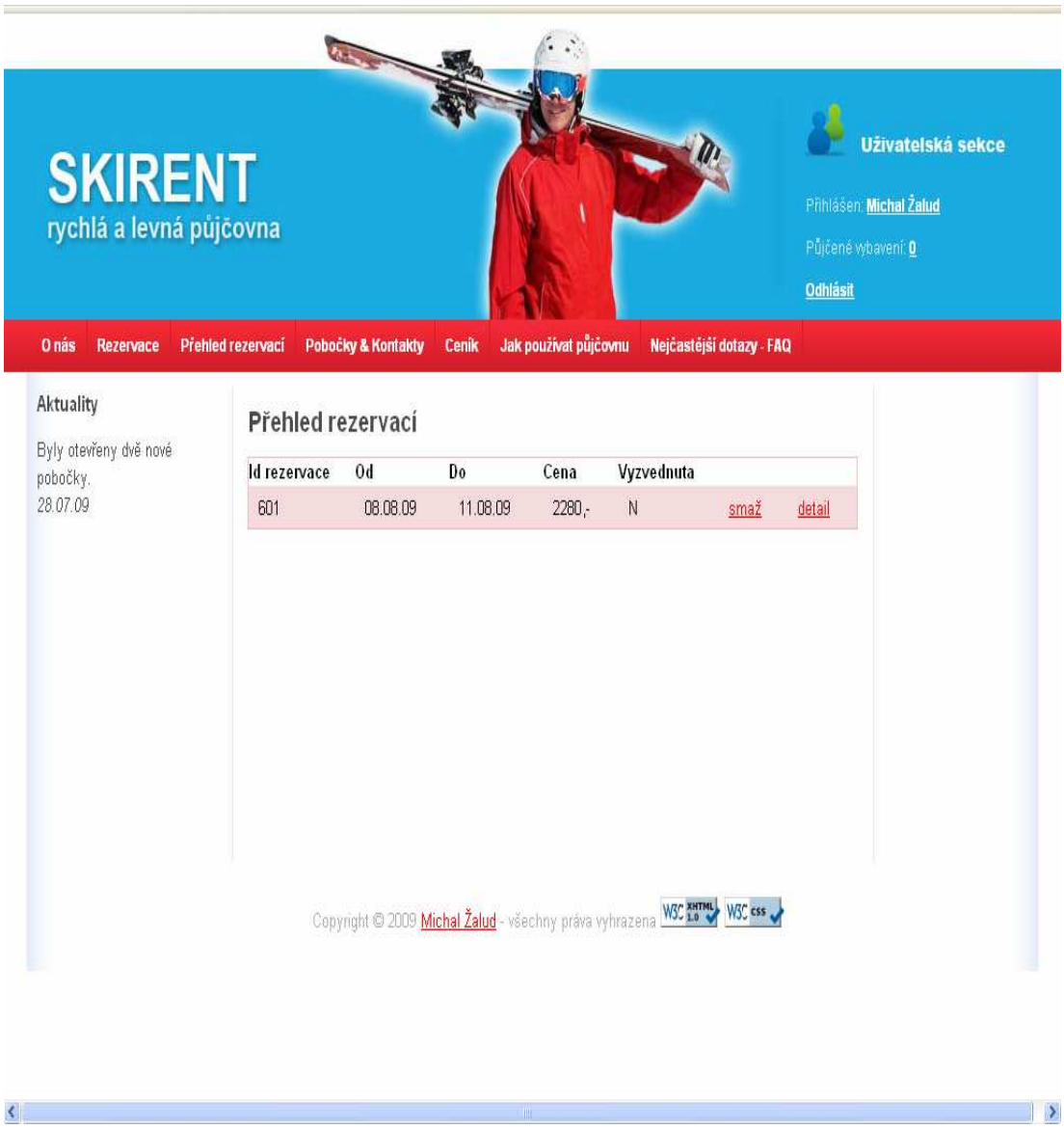
Rozvržení layoutu bylo zvoleno s přihlédnutím k co nejjednodušímu používání aplikace. Stránka je rozdělena do několika částí. Jedná se o hlavičku, menu, obsah, jenž je rozdělen na tři části – levou, prostřední a pravou, celá stránky je zakončena nezbytnou patičkou.

V hlavičce se nachází logo společnosti, které zároveň slouží jako odkaz k návratu na hlavní stránku. Vedle loga je nezbytná identifikace půjčovny zimního vybavení, a sice obrázek lyžaře. V pravé části hlavičky je umístěna uživatelská sekce, v té se po přihlášení vyskytuje jméno a příjmení přihlášeného uživatele, pod tímto identifikátorem jsou umístěny informace lišící se dle role přihlášeného uživatele. U zákazníka se jedná o počet aktuálně vypůjčeného vybavení s odkazem na přehled rezervací. Pokud je přihlášen zaměstnanec pobočky, najde se informace o celkovém počtu půjčeného vybavení s odkazem na přehled vybavení. Opravář má k dispozici informace o počtu vybavení, jenž je potřeba opravit s odkazem na jeho přehled. Při přihlášení správce je uveden počet registrovaných zákazníku s odkazem na jejich přehled. Další část uživatelské sekce je již společná jedná se o odkaz pro odhlášení.

Největší část zabírá obsah, ten je rozdělen na tří části. V té největší prostřední se nachází samotná vybraná stránka. Levá část slouží k informování o aktuálním dění, jsou zde tedy umístěny již zmíněné aktuality. V pravé části se nachází formulář pro přihlášení uživatele. Pokud je již uživatel přihlášen a nejedná se správce, je místo nevyužito, v budoucnu zde mohou být například aktuální informace o počasí. Pokud se jedná o správce, jsou zde umístěny odkazy pro změnu role.

Poslední částí stránky je patička ve které se nachází copyright a informace o validitě stránek.

.



obr. 6 - layout aplikace

7. Návrh databáze

Při uchovávání velkého množství informací je vhodné ukládat je do databáze. Já jsem pro svou práci využil relační databáze. *Relační databáze je sada nástrojů pro efektivní a spolehlivé ukládání dat a pro manipulaci s nimi*[1], data jsou uchovávaná v tabulkách, kde řádek odpovídá záznamu a sloupec atributům určitého datového typu. Vazby mezi tabulkami se nazývají relace a ty v podstatě popisují vztahy mezi objekty v reálném světě.

7.1. Fáze návrhu databáze

Návrh databáze je prvotně založen na požadavcích zadavatele, od kterého je potřeba přesně zjistit, jaké informace bude chtít uchovávat. Velice užitečné je při zjišťování požadavků zadavatele, zjistit informace i od budoucích uživatelů aplikace, což vede k minimalizaci budoucích úprav. Po získání informací se vytvoří konceptuální model, který má za úkol znázorňovat pohled člověka na danou situaci.

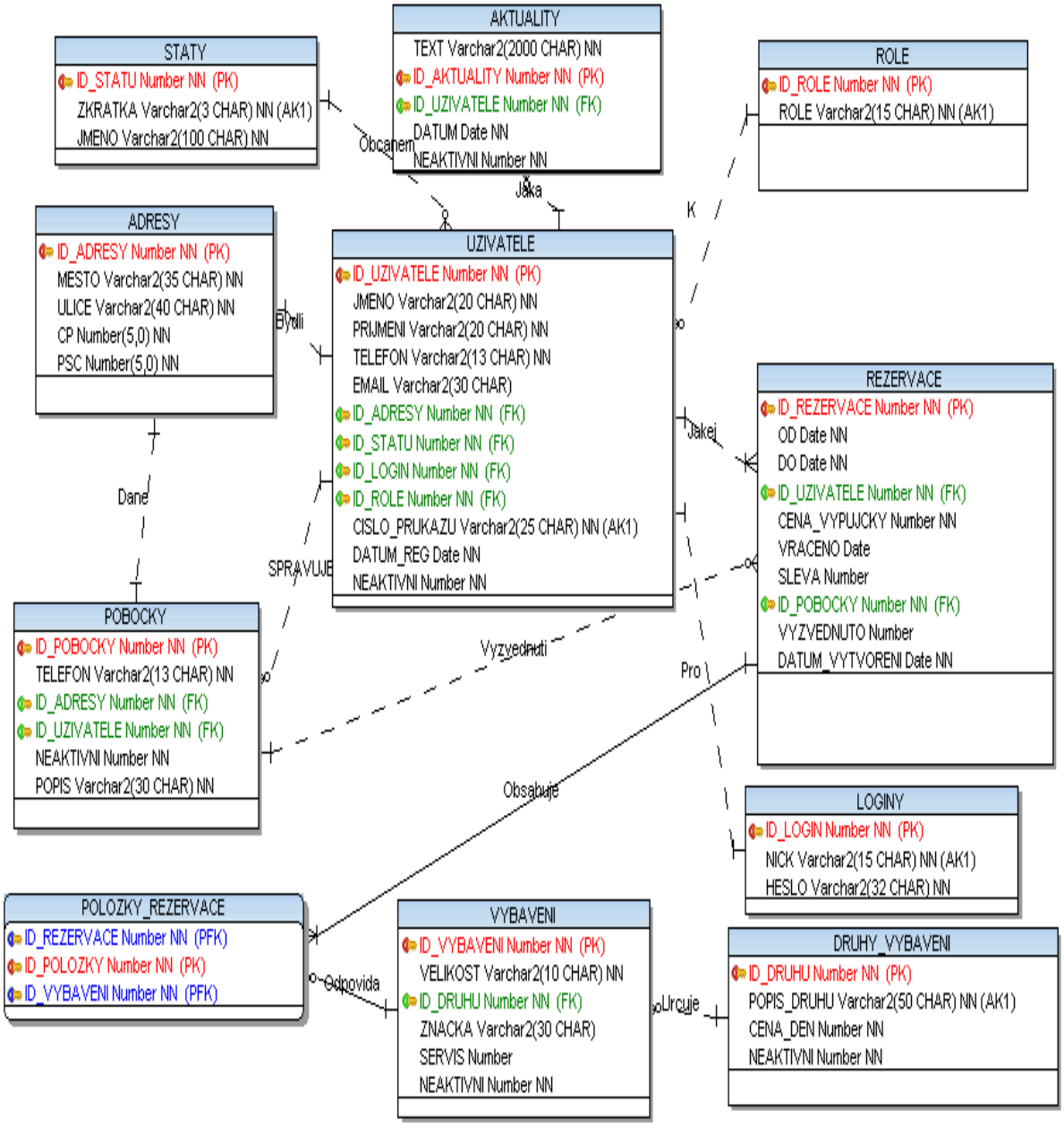
Další krok je založen na přenesení získaných poznatků do podoby relační databáze. Základem relační databáze jsou tabulky a relace mezi nimi. Tabulky je vhodné normalizovat, což lze chápat jako zjednodušení databáze vedoucí k odstranění redundance dat a efektivnímu ukládání dat. Pro tabulky platí normální formy, které jsou číslovány od 0 do 5 a dá se říci, že čím vyšší normu tabulka splňuje, tím je tabulka kvalitněji navržena.

Tabulka je v nulté normální formě, jestliže obsahuje alespoň jedno pole, jenž není atomické, obsahuje více než jednu hodnotu. První normální formu tabulka splňuje, když neobsahuje žádné nedělitelné pole, všechna pole jsou atomická. Pro druhou normální formu platí, že je v 1NF a každý sloupec, mimo primárního klíče, je závislý na primárním klíči. Platí doporučení využívat tabulky patřící minimálně do třetí normální formy, která říká, že tabulka musí patřit minimálně do 2NF a zároveň neexistuje závislost neklíčových sloupců tabulky. Aby tabulka patřila do 4NF, musí patřit do 3NF a může popisovat pouze jeden fakt nebo souvislost. Tabulka patřící do 5NF musí patřit do 4NF a není možné do ní přidat nový sloupec aniž by se tabulka rozpadla na dílčí tabulky. Při navrhování databáze se doporučuje využívat tabulek, jenž splňují minimálně 3. normální formu.

Návrh databáze mi zabral asi nejvíce času při vytváření aplikace. Ne že by byl příliš časově náročný, ale v průběhu vytváření aplikace se objevily nedostatky, proto byla potřeba návrh předělat. S předěláním databáze souvisí i jisté úpravy ve vytvářených dotazech na databázi, které byly prováděny pomocí jazyka SQL.

7.2. E-R diagram

Pro návrh své databáze jsem použil program Toad Data Modeler 3, v tomto programu jsem vytvořil datový model, pomocí tohoto programu jsem si také vygeneroval DDL skript, který mi posloužil pro vytvoření tabulek, jejich atributů a vztahů mezi tabulkami.



obr. 7 - Fyzický datový model

7.3. Stručný popis tabulek

Tabulky databáze byly navrhovány s cílem dosažení minimálně třetí normální formy. Pokud není uvedeno jinak, jsou tabulky ve třetí normální formě.

Uzivatele – v této tabulce se uchovávají data o uživatelích. Tabulka uchovává jméno, příjmení, telefon a email uživatele. Pro půjčování je nezbytné uchovávat informace o číslu průkazu, dále tabulka obsahuje informace o datu registrace. Atribut neaktivni označuje aktivitu uživatele. Tabulka je svázána s tabulkami adresy, státy, loginy. Pomocí nichž se uchovávají informace o adrese, státní příslušnosti a přihlašovacích údajích.

Role – uchovává oprávněně - uživatelské role

Staty – tabulka státy shromažďuje informace o státech, jejich názvy a zkratky

Adresy – obsahuje informace o adresách poboček a uživatelů. Adresa zahrnuje město, ulici, číslo popisné, PSČ. Jelikož existuje závislost mezi městem a směrovacím číslem tabulka nepatří do třetí normální formy.

Rezervace – slouží k uchovávání informací o rezervacích. Obsahuje nezbytné datové údaje, odkdy do kdy bude zboží půjčeno a kdy byla rezervace vytvořena a kdy vrácena. Také uchovává stav, zda byla vůbec vyzvednuta, celkovou cenu a případnou uplatněnou slevu. Tabulka je propojena s tabulkou uzivatele, tím se identifikuje osoba vlastnící rezervaci.

Polozky_rezervace – tabulka je propojena s tabulkou rezervace a obsahuje konkrétní položky rezervace, tedy konkrétní vybavení identifikované jeho číslem.

Druhy_vybaveni – tabulka obsahuje druhy půjčovaného vybavení, jejich stručný popis, cenu a příznak aktivity.

Vybaveni – slouží k uchování informací o vybavení, velikost, značku, potřebu opravy a příznak aktivity. Tabulka je propojena s tabulkou druhy_vybaveni.

Aktuality – tabulka informuje o aktuálním dění ve středisku či probíhajících akcích. Obsahuje stručný text příznak viditelnosti a datum vytvoření aktuality. Aby se vědělo, kdo danou aktualitu vložil, je propojena s tabulkou uzivatele.

Pobocky – uchovává údaje o pobočkách, jejich telefonní číslo,příznak aktivity, a díky spojení s tabulkou uzivatele a adresy, taky informace o vedoucím a adrese pobočky.

Loginy – shromažďuje přihlašovací údaje uživatelů, nick a heslo, to je uloženo zašifrované algoritmem MD5.

7.4. Sekvence

Při vkládání záznamů do tabulek, kde se používá umělého primárného klíče, je potřeba zajistit inkrementaci toho jedinečného identifikátoru. Databáze Oracle nám k tomuto účelu nabízí objekt sekvence, ta automaticky generuje čísla, ta se obvykle zvyšují, ale mohou se i snižovat. Pro ukázku uvádím příklad sekvence sloužící k inkrementaci primárního klíče v tabulce Adresy.

```
CREATE SEQUENCE ADRESYSEQ
MINVALUE 1
MAXVALUE 999999999999
INCREMENT BY 1
START WITH 1
CACHE 20
NOORDER
NOCYCLE ;
```

7.5. Triggery

Trigger je speciální procedura, množina příkazů, která se vykoná při určité operaci. Může jít o operaci delete, update či insert. Také lze definovat, jestli se má trigger spustit před nebo po dané operaci (události). Jelikož využívám umělého identifikátoru (čísla), našel trigger uplatnění při vkládání záznamu do všech tabulek databáze.

K jeho aktivaci dojde před vložením nového záznamu. K určení jedinečného identifikátoru slouží sekvence, která se inkrementuje a tigger se stará o vložení její hodnoty do tabulky.

```
create TRIGGER AdresyTrigger
BEFORE INSERT ON adresy
FOR EACH ROW
BEGIN
SELECT AdresySeq.nextval INTO :new.Id_adresy FROM DUAL;
END;
```

Triggery jsou dále využity pro zachování referenční integrity. Referenční integritu si lze představit jako nástroj sloužící k zachování vtahů mezi propojenými tabulkami. Pokud by byla porušena je obvykle vyvolána chyba. Proto používám trigger, které se postarají o zachování referenční integrity.

Příklad udává trigger používaný při situaci, že si klient rozmyslel rezervaci a chce ji zrušit. Všechny záznamy o rezervaci se tedy musí z databáze smazat. Nejedná se jen o údaje v tabulce rezervace, nutné je též odstranit záznamy z tabulky polozky_rezervace.

```
CREATE OR REPLACE TRIGGER odstraneni_rezervace
BEFORE DELETE ON rezervace
FOR EACH ROW
BEGIN
DELETE FROM polozky_rezervace
WHERE id_rezervace=:OLD.id_rezervace;
END;
```

7.6. Procedury

Procedura je databázový objekt obsahující kód, jenž se po jejím spuštění provede. Procedura pracuje s daty uloženými v databázi. Já jsem procedury využil například v situaci, kdy je potřeba upravit cenu u druhu vybavení.

Kód sloužící k vytvoření procedury, která změní cenu u druhu vybavení. Cena a konkrétní vybavení jsou proceduře předány parametrem.

```
CREATE OR REPLACE PROCEDURE zmena_ceny(cena IN NUMBER,id
IN NUMBER)AS
BEGIN
UPDATE druhy_vybaveni SET cena_den=cena
WHERE id_druhu=id;
END;
```

7.7. Funkce

Funkce v pojetí jazyka PL/SQL je posloupnost příkazů, které se vykonají po jejím spuštění a slouží k získání určitého výsledku, který následně funkce navrací. Ve své aplikaci využívám funkce ke dvěma účelům. Za prvé se jedná o získání určitých výsledků, za druhé k zjištění, zda už existuje ukládaný záznam.

Funkce pro získání výsledků jsou ve většině případů využity v klientské sekci. Každého uživatele zajímají jiné věci, proto jsou v databázi vytvořeny čtyři funkce sloužící k informování uživatelů (správce – počet zákazníků, zaměstnanec – počet vypůjčeného vybavení, opravář – počet vybavení na opravu, zákazník – počet půjčeného vybavení).

Následující kód slouží k vytvoření funkce, která slouží k získání informací o půjčeném vybavení přihlášeného uživatele.

```
CREATE OR REPLACE FUNCTION vypujceno_osoba (osoba IN
NUMBER)RETURN NUMBER AS vysledek NUMBER;
BEGIN SELECT COUNT(id_vybaveni)INTO vysledek
FROM polozky_rezervace
JOIN rezervace ON
polozky_rezervace.id_rezervace=rezervace.id_rezervace
WHERE rezervace.vyzvednuto='1' AND vraceno IS NULL AND
id_uzivatele=osoba;
RETURN vysledek;
END;
```

V databázi se vedle primárních klíčů vyskytují také atributy, které musí být unikátní. Aby při vkládání záznamů do tabulek nedošlo k chybě způsobené duplicitou záznamů, používám funkce, které mi vrací počet výskytů daného záznamu. Jedná se o kontrolu nicku, jenž slouží pro přihlašování, dále se kontroluje číslo průkazu, zadávané při registraci, a také jedinečnost druhu vybavení.

Kód pro vytvoření funkce, jenž slouží k zjištění, zda existuje druh vybavení stejného názvu jako je parametr předávaný funkci.

```
CREATE OR REPLACE FUNCTION unikatnost_vybaveni (text IN
VARCHAR2)RETURN number AS existuje NUMBER;
BEGIN
SELECT (COUNT(id_druhu)) INTO existuje
FROM druhy_vybaveni
WHERE UPPER(popis_druhu) LIKE UPPER(text);
RETURN existuje;
END
```

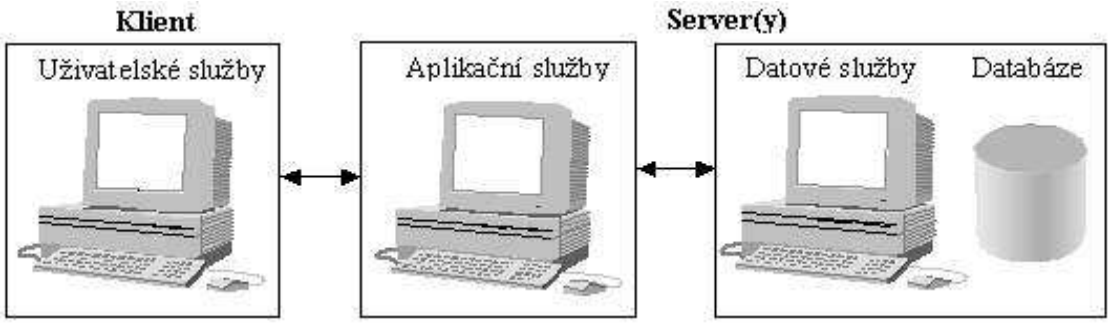
7.8. Indexy

Indexy jsou databázové objekty, jenž slouží k urychlení vyhledávání dat v databázi. Všechny použité indexy jsou vytvořeny automaticky databází. Jedná se především o primární a cizí klíče. Indexy jsou dále vytvořeny u jedinečných atributů jako je nick či číslo průkazu.

8. Použité technologie

V dnešní době, kdy prakticky každý počítač je připojen k internetu a možnost připojení je už prakticky na každém místě a každý operační systém disponuje webovým prohlížečem, se s velkou pompézností rozvíjí využívání tzv. webových aplikací. Webový prohlížeč se využívá jako klient a není tedy potřeba instalovat žádný dodatečný uživatelský software. Při využití prohlížeče jako klienta hovoříme o něm jako o tenkém klientovi, jde o to, že nezná nic z logiky aplikace.

Z důvodu využitelnosti pro více aplikací se dá o prohlížeči hovořit jako o univerzálním klientovi. Prohlížeč tedy funguje jako prezentační vrstva.Logická vrstva, jak už sám název napovídá, má za úkol spravování logiky celé aplikace, také se stará o komunikaci mezi prezenční a datovou vrstvou. Mezi prostředky logické vrstvy, je možné ji nazývat i prostřední vrstvou, lze zařadit např. jazyk PHP v součinnosti s webovým serverem. Nejrozšířenějším webovým serverem je Apache, dále se lze setkat SJSWS a dalšími. Pod pojmem datová vrstva si lze představit jakési uložiště dat, tedy databázi respektive databázový server, pod kterým databáze běží.



obr. 8 - třívrstvý model (4)

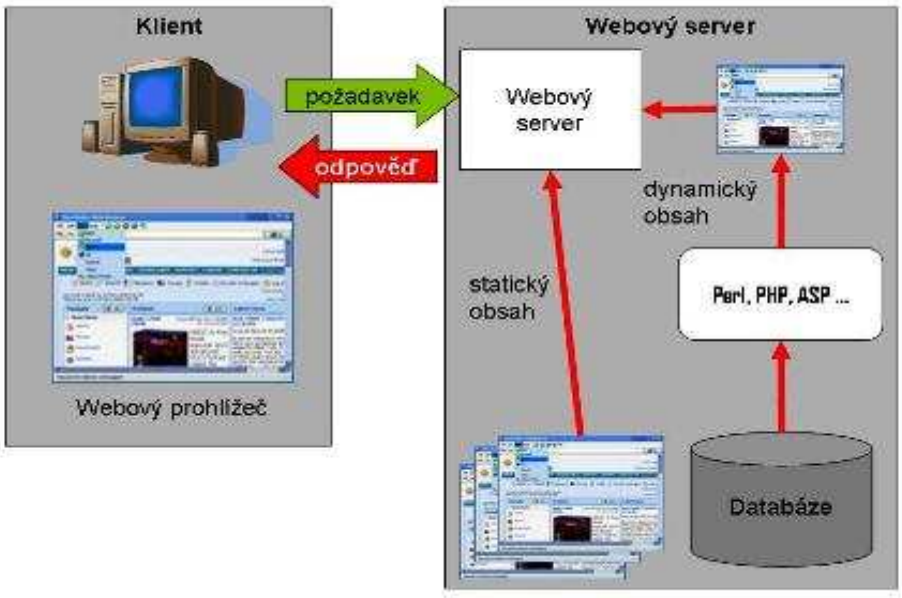
8.1. Webový prohlížeč

O webovém prohlížeči se dá říci, že je to program mezi jehož hlavní funkce patří komunikace s HTTP serverem, od kterého přijme kód a následně ho zpracuje. Přijatý kód může být v několika jazycích, např. HTML, XHTML, prohlížeč z komunikace s webovým serverem pozná o jaký jazyk jde a podle toho zformátuje a zobrazí webovou stránku. Mezi nejznámější prohlížeče patří Windows Internet Explorer, Mozilla Firefox či Opera.

8.2. Webový sever

Hlavním úkolem webového servu je odpovídat na požadavky ze strany klienta. Webové servery mohou být hardwarové nebo softwarové. Jedná-li se o hardwarové, jde o servery umístěné kdekoliv ve světě, na které klient zasílá požadavky pomocí internetu a stejnou cestou mu server odpovídá. V případě softwarového serveru jde o program běžícím pod použitým operačním systém a mluvíme o něm jako o démonu.

Mezi nejpoužívanější patří Apache HTTP Server, který jsem použil i já při vývoji své aplikace. Jeho velkou předností je otevřený zdrojový kód a podpora mnoha platforem, např. Linux, Microsoft Windows.



obr. 9 - komunikace klient-server (6)

8.3. Databázový server

Databázový server je *soubor programových prostředků určených pro práci s daty[1]*, ta také zprostředkovává klientům. Dnes se většinou používají relační databázové servery. Pro komunikaci je využíván databázový jazyk SQL, komunikace vypadá tak, že klient vznese na server požadavek v jazyce SQL a server odpoví většinou množinou dat. Mezi nejrozšířenější servery patří MySQL, či produkty firmy Oracle.

Já jsem svoji databázi "postavil" na databázovém serveru Oracle XE (Expres Edition), jde o volně dostupnou verzi, která je *postavená na základu Oracle Database 10g Release 2 a je plně kompatibilní se všemi ostatními edicemi Oracle Database.* K omezením této verze patří např. využívání jen jednoho jádra vícejádrového procesoru.

8.3.1. Příkazy pro práci s databází

Pro vznášení požadavků na databázi je vhodné si vytvořit funkce složené z jednotlivých příkazů pro práci s databází. Vhodné je to především z důvodu ušetření času a hlavně přehlednosti. Nejlépe je to vidět na mnou nejpoužívanější funkci dotaz, která slouží k vznášení dotazu na databáze. Místo několika řádků, sloužících k připojení a vykonání dotazu, zavolám funkci s parametrem, který udává konkrétní SQL kód - požadavek. Po vykonání příslušného dotazu, je výsledek následně předán jako návratová hodnota.

```
function dotaz($sql) {
$c = oci_connect ('jmeno','heslo','//localhost/XE',
'utf8');
if(!$c){
$e=oci_error();
return $e['message'];
exit();
}
else{
$s = oci_parse($c, $sql);
oci_execute($s);
oci_fetch_all($s,$res);
oci_close($c);
return $res;}
}
```

Oci_connect slouží k připojení k databázi, jako první parametr se udává uživatelské jméno, následuje nezbytné heslo, třetí parametr slouží pro určení databáze k připojení, vhodné je také určit kódování.

Oci_error vrací případné chybové hlášení. K analyzování SQL dotazu slouží příkaz *oci_parse*, až po analyzování je možné vykonat dotaz pomocí příkazu *oci_execute*. Získání výsledků dotazu zajistí *oci_fetch_all*. Vykonání dotazu se provádí pomocí kurzoru , který se zavře pomocí *oci_close*.

8.4. Skriptovací jazyk PHP

PHP je skriptovací programový jazyk, určený převážně pro programování dynamických webových stránek. PHP skripty bývají prováděny především na straně serveru, tudíž je k uživateli přenesen až výsledek, tedy HTML kód, jenž webový prohlížeč zobrazí.

8.5. TinyMCE editor

Možnost informovat zákazníky případně i zaměstnance o novinkách či aktuálně probíhajících akcích je velice užitečná věc. Tímto právem disponují správci poboček. K tomuto účelu jsem využil editor zobrazovaného obsahu, takzvaný WYSIWYG editor, dostupných jich je celá řada, já jsem využil volně dostupný TinyMCE editor.

9. Ošetření vstupů od uživatele

Kontrola uživatelem zadávaných dat je nesmírně důležitou součásti ochrany aplikace, protože velké množství útoků je vedeno právě touto cestou a útočníkovi k nim většinou stačí práva běžného uživatele. Mezi zdroje možného útoku lze zařadit:

- obsah formulářových polí
- URL adresy požadavků
- cookies
- HTTP hlavičky

9.1. Třída pro kontrolu formulářů

Pro kontrolu formulářů jsem si vytvořil třídu, která kontroluje správnost zadaných údajů od uživatele. Ve třídě jsou definovány pravidla pro kontrolu a konverzi.

Konverze se využívá pro odstranění bílých znaků ze začátku a konce řetězce. Je možno využít i další konverze jako například konverze využívající regulárních výrazů.

Pravidel pro kontrolu údajů od uživatele jsem vytvořil celkem sedm. Jedná se o pravidlo `is_fill`, to slouží k zjištění, zda bylo dané pole vyplněno. Ke kontrole správného zadání emailu jsem použil pravidlo `is_email`, využívající k tomu určený regulární výraz. Regulárních výrazů jsem využil i při kontrole čísla popisného (`is_cp`), PSČ (`is_psc`), čísla průkazu (`is_prukaz`), zde je nutno dodat, že formát čísla průkazu je nutno vyřešit dle konkrétních požadavku při případném užití v praxi, jelikož je mnoho formátu identifikačních čísel. Nutnost zadání spočívá hlavně v dokonalé identifikaci uživatele, v praxi jsem se setkal i s nutností zadat dokonce dva průkazy. Také je nutné kontrolovat správný formát telefonního čísla, zvolil jsem mezinárodní formát čísla, ten má tvar například +420123456789, ke kontrole jsem opět využil regulárního výrazu.

Třída také obsahuje pravidlo pro kontrolu hesel. Je standardem heslo zadávat dvakrát, jednou jako samotné zadání, podruhé pro kontrolu a to z důvodu případného překlepu.

Třída má jednu funkci *function uzivatelska_data($idata)*, ta slouží k předání hodnoty z proměnné *$_POST*, které je naplněna odeslanými hodnotami formuláře, ten je odeslán po vykonání metody *POST*.

9.1.1. Použití třídy

Použití třídy spočívá ve vytvoření instance třídy a nadefinování pravidel pro kontrolu, která se mají na konkrétní položku použít.Na položku formuláře lze aplikovat jak konverze tak také jedna z možností pro kontrolu.

Například pomocí PHP funkce trim se zbavit bílých znaků na začátku a na konci řetězce a poté kontrolovat položku zda je vyplněná.

```
$kontrola = new form_kontrola; // vytvori instanci tridy
$kontrola->rules = array(      // nadefinuje pravidla
  'jmeno' =>array(
  'convert' => 'trim',
  'check' => 'is_fill'
  ),
...
);
if (isset($_POST['save'])){
$kontrola->uzivatelska_data($_POST);// data se převezmou
};                                    // .. z pole $_POST
```

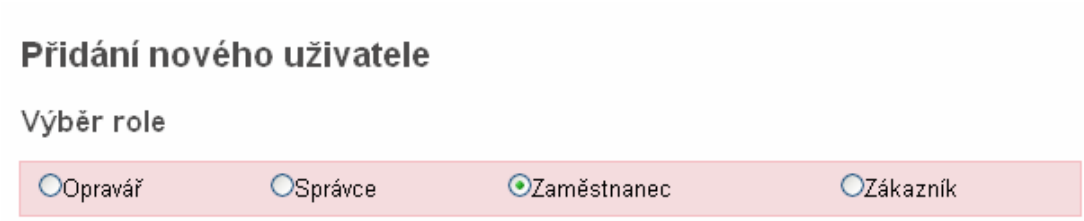
10. Popis aplikace

10.1. Registrace

Registrovat se musí každý uživatel aplikace. Zákazník se musí registrovat, aby si mohl rezervovat vybavení. Ostatní uživatele zanáší do systému (registruje) vedoucí pobočky. Jak na zákazníka tak na vedoucího pobočky čeká vyplnění registračního formuláře. Aplikace kontroluje správnost zadávaných dat, jako je například tvar emailu, telefonního čísla či jedinečnost přihlašovacího jména.

obr. 10 - registrace

Jelikož musí být správce pobočky schopný přidávat do systému zaměstnance, je nutno ještě před registrací vybrat pozici, na kterou je zaměstnanec přijímán.



obr. 11 - registrace výběr role

Může se stát, že uživatel zapomene své přihlašovací údaje. V takovém případě je nutné kontaktovat jednoho ze správců poboček. Správce má právo zasahovat do registračních údajů všech uživatelů, tudíž může zadat nové přihlašovací jméno a heslo. Oznámí je uživateli a následně mu doporučí změnění údajů.

10.2. Rezervace a vypůjčení na pobočce

Rezervovat vybavení mohou registrovaní zákazníci. Prvním krokem k vytvoření rezervace je určení, odkdy dokdy si zákazník vybavení rezervuje. Výběr probíhá pomocí přehledného kalendáře, nejdříve se zvolí počáteční den a poté konečný den rezervace. Dále je nutné vybrat jednu z několika poboček, kde si vybavení zákazník vyzvedne. Následuje samotný výběr požadovaného vybavení. Po potvrzení rezervace je možné vytisknout fakturační údaje. Rezervovat si vybavení předem má pro zákazníky výhodu slevy ve výši 5% z celkové ceny rezervace.

obr. 12 - kalendář

Pokud rezervaci vytváří zaměstnanec na pobočce, postupuje úplně stejně jako zákazník, jediný rozdíl jev tom, že jako úplně první musí od zákazníka získat osobní údaje, aby se vědělo, komu je rezervace určená. Po potvrzení rezervace musí zaměstnanec dvakrát vytisknout fakturu. Jedna je určená pro půjčovnu a jedna pro zákazníka.

Pokud zaměstnanec vyřizuje půjčení vybavení přímo na pobočce, postupuje úplně stejně jako při vytváření rezervace, s tím že jako počátek rezervace vybere aktuální den.

10.3. Vyzvednutí a vrácení vybavení

Vyzvednutí i navrácení vybavení musí být provedeno jako celková rezervace. Je tedy nutné vyzvednout a navrátit najednou všechny položky rezervace.

10.3.1. Vyzvednutí vybavení

Zaměstnanec je povinen zadat do systému skutečnost, že vydal určité rezervované vybavení. Celý proces probíhá velice intuitivně, pomocí tabulkového výpisu vybere zaměstnanec konkrétní rezervaci, jedním kliknutím potvrdí její vyzvednutí. Také si může zobrazit detail dané rezervace opět pouze jedním kliknutím na číslo rezervace. Nutné je podotknout, že se zaměstnanci vypisují pouze rezervace, určené k vyzvednutí v aktuálním dnu.

obr. 13 - vyzvednutí rezervace

10.3.2. Vrácení vybavení

Vrácení vybavení je stejně jako vyzvednutí velice intuitivní a patří do kompetence zaměstnanců. Nejdříve je zapotřebí vybrat konkrétní rezervaci, která se navrací. Pomocí zaškrtávacích políček se vyberou vybavení, která potřebují opravit. Poté už se jen potvrdí stisknutím tlačítka Vrácení. Do systému se tím zanese navrácení vybavení a případná nutnost opravy.

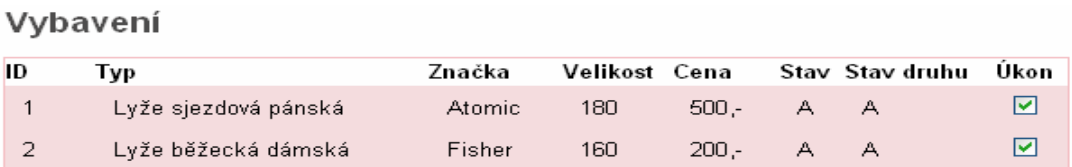
obr. 14 - vrácení vybavení

10.4. Administrativní část

Administrativní část slouží správci pobočky k úpravám a zadávání nových poboček, aktualit, uživatelů a samozřejmě vybavení.

10.4.1. Úprava vybavení

Upravovat lze buď konkrétní vybavení nebo celý druh vybavení. U práce s konkrétním vybavením se pomocí zaškrtávacího políčka vybere vybavení a poté pomocí tlačítka požadovaný úkon (aktivovat, deaktivovat, odstranit).



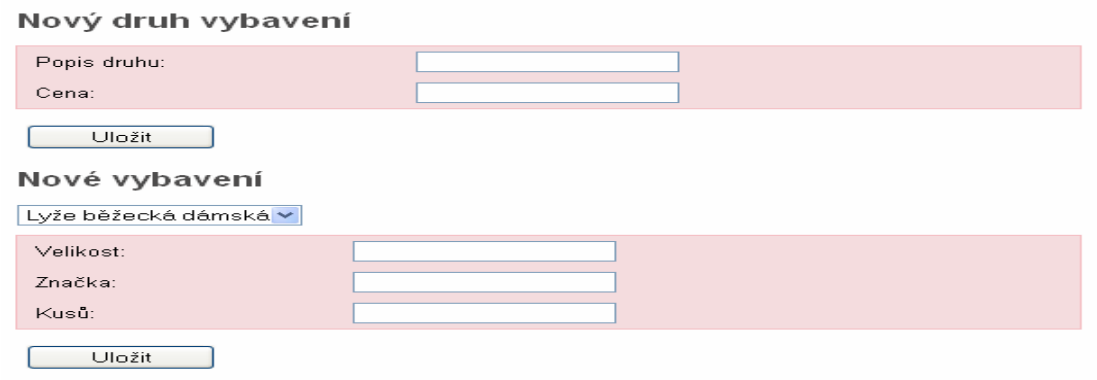
obr. 15 - úprava vybavení

Při úpravě druhu lze mimo výše zmíněných úkonu také upravovat cenu za vypůjčení.

obr. 16 - úprava druhu vybavení

10.4.2. Nové vybavení

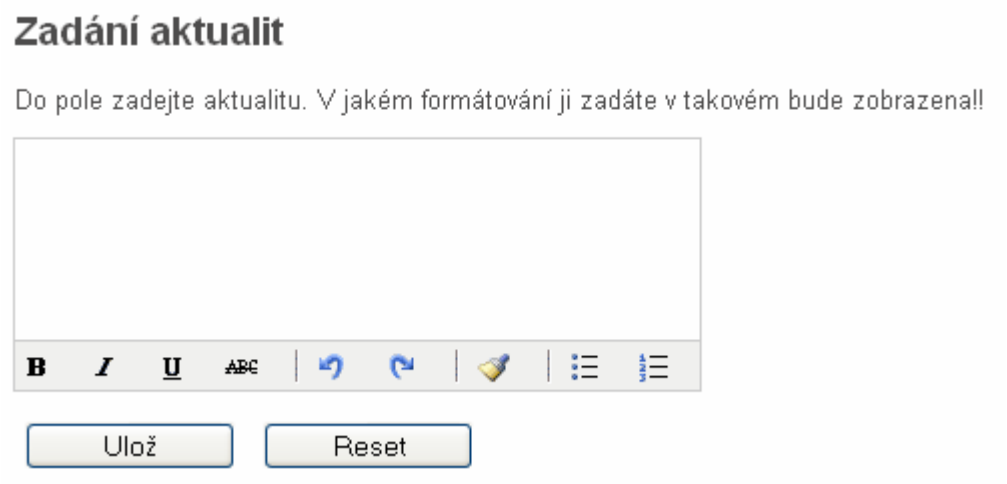
Před zadáním vybavení je nutné nejdříve vybrat druh vybavení, nezbytné je také uvést značku a počet kusů. V případě, že ještě neexistuje požadovaný druh, je nutné ho nejdříve zavést do systému.



obr. 17 - zadání nové vybavení

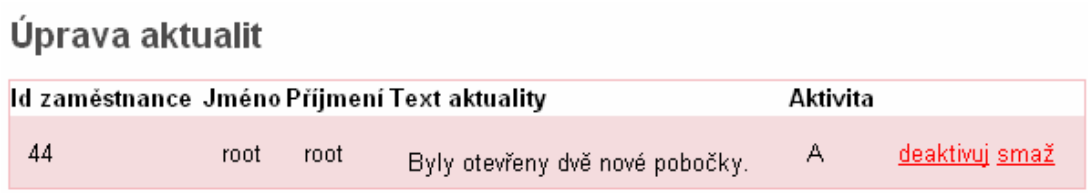
10.4.3. Úprava a zadání aktualit

Zadaní aktualit je řešeno pomocí WYSIWYG editor, tedy jak se text zadá v takovém formátu bude následně zobrazován.



obr. 18 - zadání aktuality

Aktuality lze aktivovat a deaktivovat. Samozřejmě je správce může i úplně smazat. Vše je řešeno pomocí jednoduchých odkazů.



obr. 19 - úprava aktuality

10.4.4. Úprava a zadání poboček

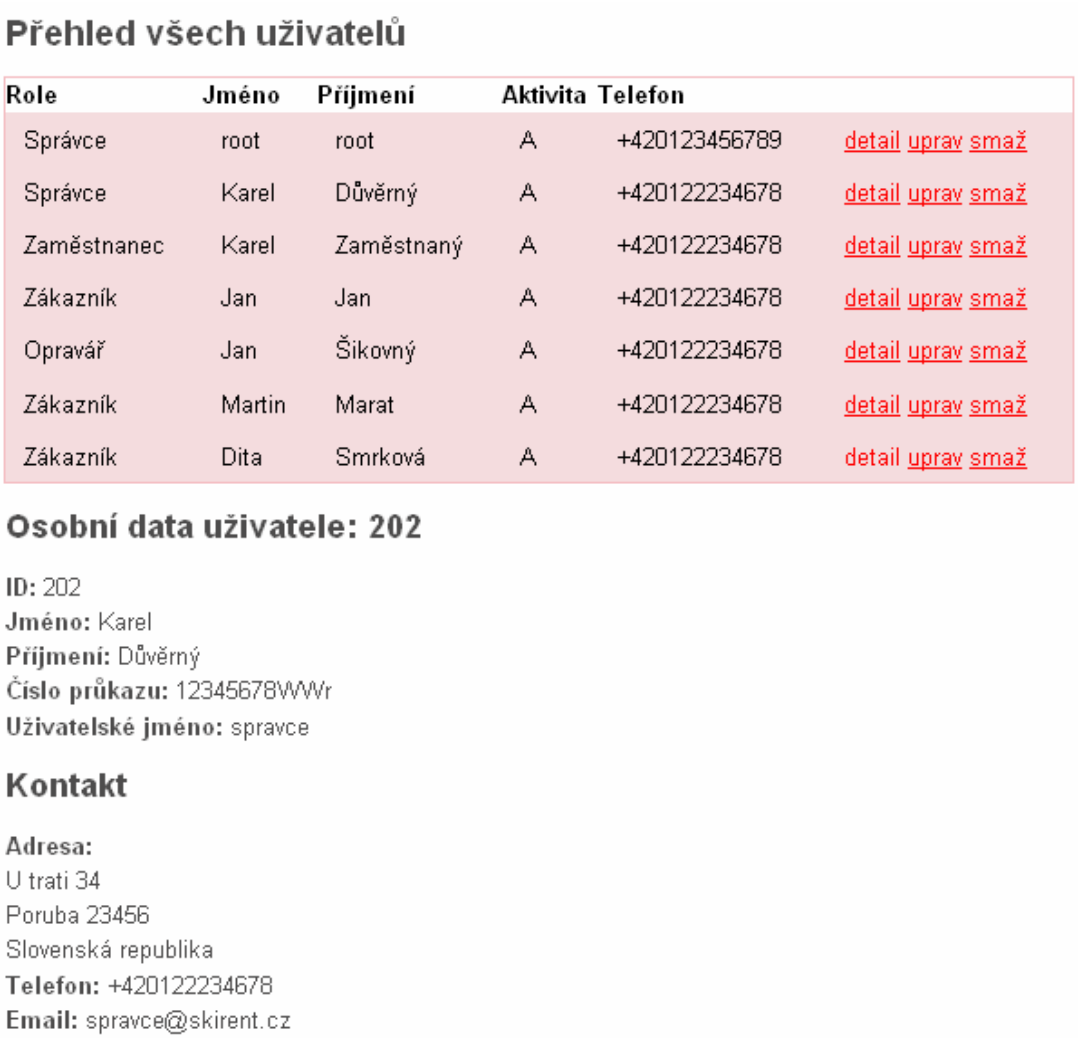
U pobočky lze upravovat veškeré údaje (adresa, popis,vedoucí a stav), asi nejčastěji se budou upravovat informace o stavu, tedy zda je pobočka otevřena či uzavřena.

Pro zadání nové pobočky je nutné vyplnit obdobný formulář jako je uveden na obr. 20.

obr. 20 - úprava pobočky

10.5. Spravování uživatelů

Správce pobočky má možnost spravovat všechny uživatele aplikace. Může u uživatelů editovat registrační údaje. Správce má možnost dočasně zakázat používání aplikace všem uživatelům tím, že nastaví jejich stav na neaktivní. Při případném propuštění zaměstnance či při nevhodném chování zákazníka, může takového uživatele správce ze systému úplně smazat.



obr. 21 - přehled uživatelů

11. Závěr

Cílem mé bakalářské práce bylo vytvořit aplikaci umožňující správu zimního vybavení v malém zimním středisku. Při vytváření aplikace jsem využíval znalostí svých kamarádů, kteří v půjčovnách pracují. Nejedná se jen o jeden podnik a z toho plynou rozdílné pohledy a zkušenosti. Velice dobře je to vidět u problematiky samotného půjčování, já jsem se rozhodl využít “bezsetovou“ variantu, jedná se o půjčování konkrétního zboží na rozdíl od půjčení setu, jenž obsahuje například lyže a boty.

Při rozhodování, jakou technologii použiji, jsem vycházel ze znalostí získaných během studia. Takže zcela logicky padla volba na databázový server Oracle. Využití programovacího jazyka PHP byla také zcela jasná volba. Pro návrh databáze jsem vystačil se znalostmi načerpanými během studia. Znalosti programovacího jazyka jsem si ale musel v podstatné míře doplnit.

Aplikace obsahuje všechny požadované funkcionality, postup rezervace je logicky členěn, tudíž by zákazníci neměli mít problém při orientaci v systému. Aplikace také nabízí dostatečné prostředky k spravovaní poboček a vybavení.

V moderních aplikacích je nejdůležitější první dojem, který uživatel získá ze vzhledu aplikace. Myslím si, že mnou vyvinutá aplikace vzhledem nikterak nezaostává za moderními aplikacemi.

Použitá literatura a ostatní zdroje

[1] LACKO, Luboslav. *Oracle : Správa, programování a použití databázového systému*. Marek Kocan. 1. vyd. Brno : Computer Press a.s, 2007. 576 s. ISBN 978-80-251-1490-2.

[2] W. JASON, Gilmore. *Velká kniha PHP a MySQL 5 : kompendium, znalostí pro začátečníky i profesionály*. RNDr. Jan Pokorný. 1. vyd. Brno : Zoner Press, 2007. 864 s. ISBN 80-86815-53-6.

[3] ORACLE DATABASE : BEZPEČNOSTNÍ MECHANISMY. *ORACLE DATA SHEET* [online]. 2006 [cit. 2009-05-06].

[4] *Architektura databází* [online]. [2003] [cit. 2009-05-10]. Dostupný z WWW: <http://www.fs.vsb.cz/books/MSSQLServer/MSSQL_soubory/SQL_index3.htm>.

[5] MAJ, Artur . *Apache 2 with SSL/TLS* [online]. 2005 [cit. 2009-05-10]. Dostupný z WWW: <http://www.securityfocus.com/infocus/1818>.

[6] *Webový server* [online]. 2009 [cit. 2009-05-10]. Dostupný z WWW: < http://cs.wikipedia.org/wiki/Webový_server>.

[7] KACÁLEK, MALÝ. *Zabezpečení webových aplikací I. : klientské skriptovací jazyky* [online]. 2007 [cit. 2009-05-05]. Dostupný z WWW: <http://access.feld.cvut.cz/view.php?nazevclanku=zabezpeceni-webovych-aplikaci-i-klientske-skriptovaci-jazyky&cisloclanku=2007090001>.

[8] KACÁLEK, MALÝ. *Zabezpečení webových aplikací II. : databáze* [online]. 2007 [cit. 2009-05-05]. Dostupný z WWW: <http://access.feld.cvut.cz/view.php?nazevclanku=zabezpeceni-webovych-aplikaci-ii-databaze&cisloclanku=2007080002>.

[9] RUŽIČKA, Pavel. *Bezpečnost především : použití SSL* [online]. 2002 [cit. 2009-05-05]. Dostupný z WWW: <http://interval.cz/clanky/bezpecnost-predevsim-pouziti-ssl/>.

Příloha A Instalace

Pro funkčnost aplikace je zapotřebí nainstalovat webový sever Apache a zprovoznit PHP, například jako modul Apache. Obě technologie jsou volně dostupné. Dále je potřeba nainstalovat a nakonfigurovat databázový systém Oracle.

1. **Webový sever Apache**

   http://httpd.apache.org/download.cgi

2. **PHP**

   http://www.php.net/downloads.php

   1. a 2. Webový server Apache + PHP – balík předkonfigurovaných instalací

   http://www.apachefriends.org/en/xampp.html

3. **Databázový sever Oracle 10g Express Edition**

   http://www.oracle.com/technology/software/products/database/index.html

4. **Konfigurace webového serveru**

Po úspěšné instalaci je vhodné si vytvořit účet v databázi, toho lze dosáhnout po přihlášení se jako administrátor. V souboru function.php (umístěn je v podadresáři include) se musí zadat údaje pro připojení k databázi. Je nutné zadat přihlašovací jméno (username) a heslo (password), dle údajů zvolených při vytváření databázového účtu.

```
$c = oci_connect
('username','password','//localhost/XE','utf8');
```

5. **Vytvoření a editace databázových objektů**

SQL skript vytvoreni_db.sql (umístěn v adresáři vytvoreni) slouží k vytvoření tabulek, triggeru, sekvencí, funkcí a procedur. Skript je možno spustit několika způsoby, například z nástroj pro správu databáze Oracle 10g XE. Skript také vytvoří

správce pobočky, který může do systému zadávat další uživatele. Pro přihlášení použijete následující údaje nick – spravce1 heslo – root.

6. aplikace

Finálním krokem k zprovoznění aplikace je její překopírovaní do kořenového adresáře webového serveru. Je tedy nutné překopírovat adresář skirent z přiloženého CD do adresáře htdocs webového serveru.

7. Spuštění aplikace

Aplikace se spustí zadáním adresy http://localhost/skirent do webového prohlížeče.

Příloha B Adresářová struktura přiloženého CD

--- skirent (adresář)

------ css (adresář)	css styly aplikace
------ images (adresář)	obsahuje obrázky pro aplikaci
------ include (adresář)	kódy aplikace

--------- Opravar (adresář)

--------- Spravce (adresář)

--------- Zakaznik (adresář)

--------- Zamestnanec (adresář)

------ scripts (adresář)

--------- tiny_mce (adresář)	obsahuje TinyMCE editor

------ index.php (soubor)

------ login.php (soubor)

--- vytvoreni (adresář)

------ vytvoreni_db.sql (soubor)

--- ZaludM_pujcovna vybaveni_IR_2009.pdf (soubor)